Volume 5 | Issue 1 Article 21

2016

Instrumentation of a Self-Correcting Data Acquisition System for Ultrafast Lasers

Fabricio S. Marin
DePaul University, fabricio.s.marin@gmail.com

Instrumentation of a Self-Correcting Data Acquisition System for Ultrafast Lasers

Acknowledgements

I would like to thank Dr. Eric Landahl for providing me with the opportunity to work with him on his research, for guidance and for his continued support. I want to thank Mary Tarpley for assisting me in the laser laboratory with soldering, laser alignment, and programming. I also want to thank Joe Boesso for helping me with setting up the encoder programming. Lastly, I want to thank DePaul University's Undergraduate Summer Research Project program for funding my research and for their continued support in student research.

Instrumentation of a Self-Correcting Data Acquisition System for Ultrafast Lasers

Fabricio S. Marin*

Department of Physics

Eric Landahl, PhD; Faculty Advisor

Department of Physics

ABSTRACT Optical phenomenon in semiconductors and other light-sensitive materials typically happen at very short time durations, and require instruments capable of measuring time resolutions on the order of nanoseconds to femtoseconds. Electronics can only measure up to nanosecond-time lengths. The purpose of this summer research project is to design a system capable of achieving variable time delays with resolutions up to femtosecond range to use in time-resolved experiments with the pump-probe technique. The time delay is achieved using a delay stage driven by a micro stepper-motor which moves in variable increments while an encoder counts the steps and tells a computer to make adjustments as necessary. This requires developing a program to communicate between electronics connected to a raspberry pi and the stepper motor. Results show that the system is limited by the encoder resolution, which yields roughly 100 femtosecond resolution at 1/25th of a step and 2400 femtoseconds per full step. A relatively inexpensive and programmable setup like this can be used to study optical phenomenon such as reflectivity, absorption, transmission, or polarization in semiconductors.

INTRODUCTION

Light-sensitive materials such as semiconductors and certain proteins exhibit optical changes when excited by light. These phenomena only last for a very brief period of time after the light strikes the material, and recording these changes requires equipment capable of photo-exciting a sample and recording the material's response nanoseconds to femtoseconds later [1]. In digital, time-sensitive processes, a computer may execute a set of commands and use crystal oscillators for time-keeping purposes.

Fmarin2@*depaul.edu*
Research Completed in Summer 2015

These oscillators are found in devices like cellphones, radios, computers, wristwatches, and oscilloscopes [2]. The resolution of these crystals is proportional to their resonance frequency, which is generally in the Megahertz range [2]. For this research, we require time resolutions that fall between one nanosecond and one femtosecond. That time resolution would require a crystal oscillator with a frequency that is 7 to 9 orders of magnitude greater than the most commonly used ones. In other words, conventional methods of time keeping will not suffice. In order to work around this limitation, a different technique uses light as the clock instead of a crystal/electronic oscillator.

An alternative to mechanically-limited clocks is the pump-probe method. This method sends a pump pulse of light to excite the sample material followed by a probe pulse [3]. The second pulse of light measures optical changes such as transmission, reflection, or the polarization of the light going through the material [3]. The time delay between the first and second beam striking the sample is determined by the distances that these beams travel, or rather by the difference between the second beam's longer path and the first beam's shorter path.

The time delay (t) is calculated by dividing twice the delay distance (d) by the speed of light (c):

$$t = \frac{2d}{c} \qquad (1)$$

Distance is multiplied by two to account for the forward and backward path.

For example, if the second beam travels 30 centimeters farther than the first one, then it will arrive two nanoseconds after the first beam. Now, if the second beam is separated by one millimeter, then it will arrive 6.7 picoseconds after the first beam. Basically, the time resolution of this system is limited by how well it can measure and keep track of distances. Light has the property of being an electromagnetic wave with a velocity fixed to 3×10^{8} meters per second. This constant velocity allows for precise time measurements that far exceed that of crystal oscillators. The main purpose of this project is to design a system that takes high-precision distance measurements for a range of distances in an automated fashion to be used with the pump-probe technique.

METHODS

Pump-Probe Setup

In Figure 1, an infrared laser beam is directed through a series of optics and is separated into two beams using a beam splitter. One beam will serve as the pump, and the other, as the probe. The beam of higher intensity proceeds in the direction of the sample material, while the lower intensity probe beam takes a detour of length $2d$, but eventually reaches the sample material at a later time, t, given by Equation 1. The delay distance is controlled mechanically using a platform holding a mirror that glides back and forth on steel rods. Attached to this delay-stage setup is a timing belt held by two stepper motors; one which moves the belt, and the other, which monitors the position of the delay stage using an optical encoder mounted onto the motor.

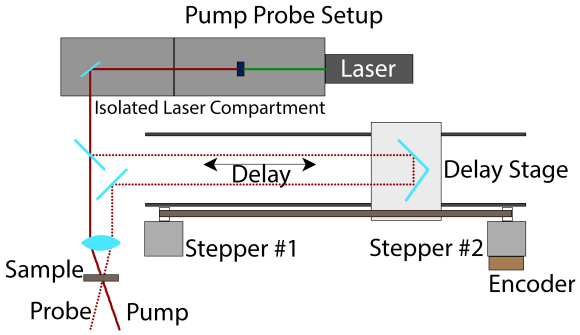

Figure 1: Simplified Pump-Probe setup.

Stepper Motors

Stepper motors are different from DC motors since they do not run on a constant voltage supply. They instead move in small increments when current is applied to one of the motor's four (or more) coils. The motors work by creating an induced magnetic field when current is applied to the coils. This magnetic field interacts with the magnetic field of the center

shaft creating torque which causes it to align with the induced field. Once aligned, current is applied to an adjacent coil so that a new magnetic field causes the center shaft to align with that coil. This process is repeated, and each time the center shaft aligns from one coil to another, this is considered one full step. It is possible to achieve fractional increments by applying current to multiple wires simultaneously, allowing for increased time resolution. Two NEMA (National Electrical Manufacturers Association) 17 stepper motors are used in this setup. The number 17 refers to the dimension of the cube-shaped motor where each side has a length of 1.7 inches. Each motor has four wires connected to four coils, and the number of coils determines the number of input signal possibilities. Table 1 shows the algorithm for a full-step cycle where 0 represents zero current, 1 represents a non-zero current and *A, A', B, B'* correspond to four different wires/coils. Increments other than full steps are implemented differently. The half-stepping algorithm is included in Table 2. The intermediate step between two wires shows how the current value in a way "transitions" from one coil to the next. Half stepping requires twice as many steps than full stepping and each smaller microstep, say quarter stepping, would require even more steps. Coding all the steps into software would be tedious and impractical.

A much smoother transition is accomplished by "tricking" the motors into thinking that both wires are on. Step 1-2 in the half stepping algorithm, for example, shows both coils *A* and *B* having a current running through them. The same result is accomplished by giving coils *A* and *B* opposite values, then running a high frequency alternating current of 0V to 5V through both coils. To the motor, both coils will appear to be on. Now if coil *A* is on only a quarter of the time, then by contrast coil *B* will be on three quarters of the time. Combining Pulse-Width Modulation (in software) and logic gates (in hardware) achieves different current values, even though only binary values of 1s and 0s are being used. The trick is to send pulses of 1s and 0s to one wire in the form of square waves where the signal is 1 a certain percentage of the time. Input and output signals are controlled using a computer program WiringPi [4], Octave software [5], an SPI interfacing program (spincl) [6], and a C library for the raspberry pi [7].

Table 1: Full-stepping algorithm used in program.

Table 1: Full-stepping Algorithm				
Coil	A	B	A'	B'
Step 4-1	1	0	0	0
Step 1-2	0	1	0	0
Step 2-3	0	0	1	0
Step 3-4	0	0	0	1

Table 2: Example method for implementing a half-stepping algorithm.

Table 2: Half-stepping Algorithm				
Coil	A	B	A'	B'
Step 8-1	1	0	0	0
Step 1-2	1	1	0	0
Step 2-3	0	1	0	0
Step 3-4	0	1	1	0
Step 4-5	0	0	1	0
Step 5-6	0	0	1	1
Step 6-7	0	0	0	1
Step 7-8	1	0	0	1

Self-Correcting Procedure & Calibration

After implementing a full-stepping and micro-stepping algorithm into the code, the user can call that function and tell it to move some number of steps forwards or backwards. A potential problem, if the driver chip overheats, is that the motors might miss a couple of steps in the process. In order to ensure that the delay stage moves to the desired position, a self-correcting algorithm is added into the code. The program works by instructing the encoder to return the final position value and subtracting it from the desired value. If that value is positive, then the micro-stepper missed a few steps, and it

must move forwards by the missed amount of steps. If the value is negative, then the motor over-stepped and needs to move back. This stepping and checking process is repeated until the difference between the desired position and actual position is zero. This procedure is explained visually in Figure 2.

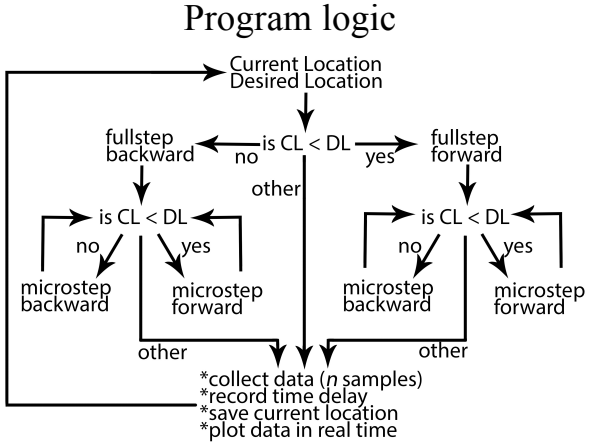

Figure 2: Flow chart describing the code logic and what it accomplishes.

The stepper motors used in this project have a 200 step/revolution resolution, or 1.8 degrees per step. The encoder counts in "pulses." The program needs to be able to convert between pulses, steps, and metric distance. To do this, the motor runs through the full-stepping cycle continuously until the timing belt does ten complete loops. If, for whatever reason, the motor misses a few steps, the belt is moved manually until the one-loop mark on the timing belt meets the second stationary mark on the steper motor shaft. During this process, the optical encoder is keeping track of the pulses. At the end of the ten loops, the encoder returns a pulse count.

The conversion factor from pulse count to metric distance is calculated by multiplying the length of the timing belt by the number of cycles it is instructed to move. This quantity is then divided by the number of pulses that the encoder counted, as shown in Equation (2):

$$mm\_per\_pulse = \frac{b*l}{p} \qquad (2)$$

where b is the length of the timing belt in milimeters, l is the number of loops the belt moves, and p is the number of pulses that the encoder value returns after l number of loops.

To convert from full-steps to a metric distance, the mm_per_pulse variable is multiplied by the pulse number that the encoder returns and then divided by the number of full-steps that the timing belt moved, as shown in Equation 3:

$$mm\_per\_fullstep = \frac{p*mm\_per\_pulse}{n} \qquad (3)$$

where p is the number of pulses, mm_per_pulse is a constant, and n is the number of full-steps.

With both conversion factors in place the user can work with metric distances, a unit more familiar than steps or pulses.

RESULTS

Once all electrical components were soldered, two stepper motors were held in place on an optical table with the timing belt on both motor's pulleys. The belt was marked with a pencil to use as a point of reference as it loops around. Hovering over that mark was one end of a paperclip which was used as a stationary point of reference indicating the beginning and end of one revolution every time the belt passed under that paperclip. The right number of steps that moved the belt one full revolution was found through trial and error. This number is still approximate and is only used to solve for the encoder value to distance conversion. After ten loops the encoder read a value of 808318 and the length of the belt was known to be 1164 mm, therefore a single encoder pulse is equivalent to 0.00144mm:

$$mm\_per\_pulse = \frac{1164\,mm}{808318\,pulses} = 0.00144\,mm/pulse \qquad (4)$$

This is the smallest resolvable distance that the encoder can read and could yield a theoretical time delay of 96 femtoseconds:

$$Time\ delay = \frac{2*1.44\,x\,10^{-6}m*10}{299792458m/s} = 96fs \qquad (5)$$

There might be other factors like mechanical limitations that make reaching this resolution unlikely.

Figures 3 and 4 show the distance traveled in 2-00 steps as a function of step increment plotted along a blue expected-values curve. Twenty distance samples are plotted for each step increment prior to implementing the self-correcting algorithm. Each of the distance samples for each step increments is off by about 0 to 3 steps, or 0 to 1.08 mm. This difference can be seen up close in Figure 4, where each of the 20 trials is represented in different colored rings. A piecewise function-like pattern can also be seen because the range of step increments is close to the encoder's resolution limit. Initially, the motor driver would overheat, causing the motor to miss 30-80 steps, or 10.8-28.8 mm. The issue was resolved by applying a heat sink to the driver chip along with a small cooling fan.

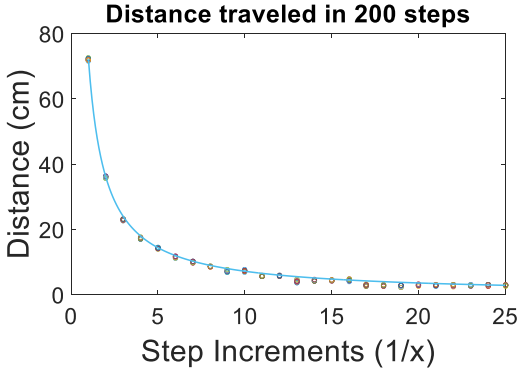

Figure 3: A full range of step incements is displayed from a full-step to a 25th of a step. Each of the 20 trials is shown in different colored rings.

In full-stepping increments, the stepper motor coils receive square-wave pulses of 5 volts. In half-stepping increments, the two active coils receive inverted values of each other, so that when one has 5V across it, the other has zero. Since the signal alternates between 5 and 0 at a very high frequency, the two coils will interpret the signal as the average of the two values, which is 2.5V. This means both induced magnetic fields from the coils will pull on the permanent magnet with equal force, causing it to align itself at a half-way point between both coils, otherwise known as a half step. Figure 5 shows the voltage value that one of the coils would register for a specified step increment. The *x*-axis shows how far the belt would move for each step.

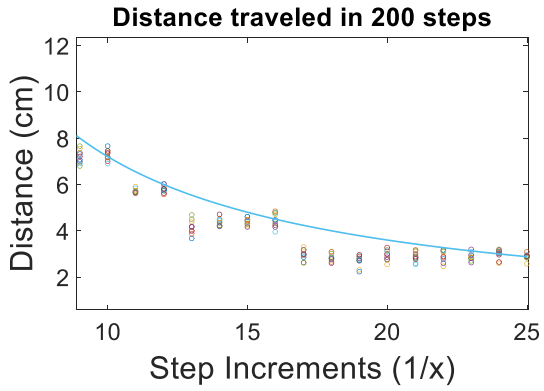

Figure 4: Encoder reaches resolution limit as seen by the stepwise-like plot over an accepted-value curve for 10th to 25th step increments. Each of the 20 trials is shown in different colored rings.

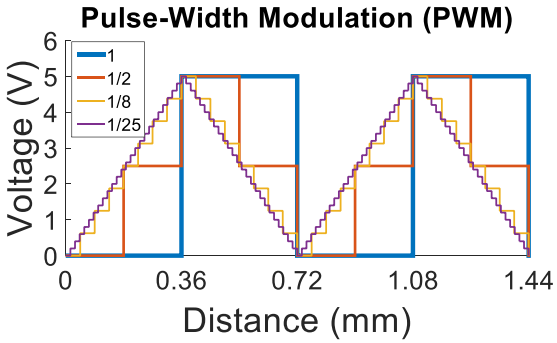

Figure 5: Full stepping shown in blue, half stepping in red, eight stepping in yellow, and twenty-fifth stepping in purple.

Stress testing the motor revealed that it is capable of 1/40th step increments, which would yield 60 femtoseconds, but as mentioned before, the encoder can only register a 25th of a step. Figure 6 shows the possible delay times that this system can achieve as a function of step increments, where 0.1 is a 10th of a step, .2 is a 5th, .5 is a half-step, etc. The smallest possible time delay was the encoder's limit, 96 femtoseconds. When the delay stage was at its maxium distance of 0.4 meters, the time delay

was calculated to be 2.7 nanoseconds using Equation 1.

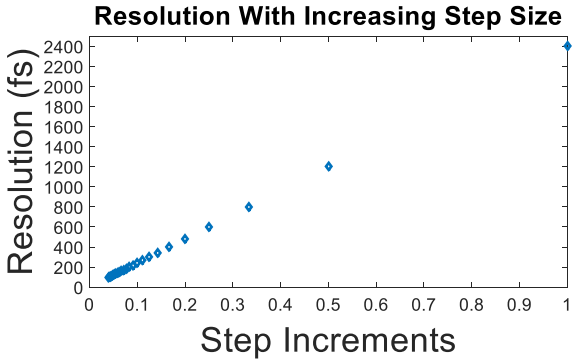

Figure 6: Resolution as a function of step increment shows final time-resolutions.

Design Limitations

Limitations were taken into account when designing this system. The cost effectiveness of the project meant that electrical and mechanical components, which were not already available, should be bought only if absolutely necessary. Other instruments could accomplish the same task of moving the delay stage but are less versatile. Instead, clever software implementation and some common circuit components were used. A micrometer apparatus fitted with gear reducers could also move the delay stage a distance 9 times smaller than the motor's smallest microstep distance of 9 micrometers. Unfortunately, the extra order of magnitude cannot be resolved by the encoder. However, there are still potential applications for this device.

The length of the timing belt length was assumed to be true according to the item's specification. For calibration purposes, it would have been better to measure the length of the timing belt with a greater resolution than that of the encoder. The micrometer would have been ideal for this situation, as its resolution is about an order of magnitude greater than the distance that the encoder can read. The micrometer would help to find a more precise distance-per-pulse value and in doing so, it would also help to find an equally precise distance-per-full-step value.

Another limitation was the motor's smallest step increment. The encoder already established a hard limit to be a 25^{th} of a full step, but assuming a higher resolution encoder was purchased, then the set up would be able to take advantage of the stepper motor's $1/40^{th}$ step increment limit. Anything less than that would not create a large enough voltage difference to generate the torque needed to overcome the pulley system's static friction.

The last limitation was an optimization problem. The pulleys used were roughly four centimeters in diameter. A larger diameter pulley would provide a larger area for the timing belt to grip onto, but it would mean that each full-step and microstep would yield a larger distance. Reducing this distance requires smaller step increments, which would reduce torque and require a higher resolution encoder. A smaller diameter pulley would yield a smaller distance per step, but would compromise grip. In the end, the initial setup was used because the stepper motor had enough torque and grip to move the delay stage without issue.

Conclusion and Potential Applications

The completion of this summer project leaves the pump-probe setup ready to be used on sample materials. Although this project mainly focused on ultrafast-laser applications, this device could be used for other purposes that require small incremental adjustments and position tracking. A telescope tracking system could benefit from this setup, as it would need to keep track of the telescope's orientation and be able to adjust it with high precision. The encoder, stepper motor, and programming provide this functionality. Another practical application of just the stepper motor, software, and encoder would be in 3D printing. Consumer 3D printers require one to three millimeters of filament extrusion resolution for a sturdy object build, and the stepper motors are capable of these resolutions.

ACKNOWLEDGEMENTS

I would like to thank Dr. Eric Landahl for providing me with the opportunity to work with him on his research, for guidance and for his continued support. I want to thank Mary Tarpley for assisting me in the laser laboratory with soldering, laser alignment, and programming. I also want to thank Joe Boesso for helping me with setting up the encoder programming. Lastly, I want to thank DePaul University's Undergraduate Summer Research Project program for funding my research and for their continued support in student research.

REFERENCES

[1] B. Moritz, T.P. Devereaux, and J.K. Freericks, *Phys. Rev. B* **81**, 165112 (2010).

[2] P. Scherz, *Practical Electronics for Inventors* (2013).

[3] V.A. Svetlichnyi, *Instrum Exp Tech* **53**, 575 (2010).

[4] G. Handerson. *WiringPi* [computer program]. (GNU, 2015). Retrieved from http://wiringpi.com/

[5] A. Weber. (2014) Octive-rpi-gpio [computer program]. Retrieved from https://github.com/octave-de/octave-rpi-gpio

[6] G. Marks, *Spincl* [computer program]. (iP Solutions, 2013). Retrieved from http://ipsolutionscorp.com/raspberry-pi-spi-utility/

[7] M. McCauley*, C Library for Broadcom BCM 2835* [Computer utilities]. (2015). Retrieved from http://www.airspayce.com/mikem/bcm2835/